Available at
www.ElsevierMathematics.com
POWERED BY SCIENCE DIRECT®
Discrete Applied Mathematics 135 (2004) 41–54
DISCRETE APPLIED MATHEMATICS
www.elsevier.com/locate/dam

Average time of computing Boolean operators ☆

A.V. Chashkin[1]

Faculty of Mechanics and Mathematics, Moscow State University, Vorob'evy Gory, Moscow 119899, Russia

Abstract

The average time of computing Boolean operators by straight-line programs of two types is studied. Upper and lower bounds for the corresponding Shannon functions are obtained. Asymptotically exact formulas for Shannon functions are derived in the case when the number of components of the operators to be computed increases with the number of their arguments.

Keywords: Average time; Straight-line programs with a conditional stop; Boolean operator

1. Introduction

The average time of computing Boolean functions by straight-line programs with a conditional stop was studied in [1]. In this paper, we consider two types of analogous programs that compute Boolean operators.

Any first-type straight-line program with a conditional stop is a sequence of operators of two types. Each operator of the first type computes a Boolean function. The arguments of that function are the values computed by preceding operators or the values of some input variables. Each operator of the second type depends on $m+1$ variables, where m is the number of program outputs, and can terminate the execution of the program. The result of a second-type operator is determined by the values computed by the program at some $m+1$ preceding steps. For each particular operator, the indices of these steps are fixed and may be different for different operators. If the last argument of a second-type operator equals unity, then the execution of the program terminates and the values of its other arguments are declared to the values of the program on

E-mail address: chash@online.ru (A.V. Chashkin).
☆ Translated from Discrete Analysis and Operations Research 5(1) (Novosibirsk, 1998) 88–103.
[1] Supported by the Russian Foundation for Fundamental Research (Grant 96-01-01068).

0166-218X/$ - see front matter
PII: S0166-218X(02)00293-7

the tuple of variables under consideration. If the last argument of the operator is zero, then the next operator of the program is executed.

Straight-line programs of the second type with a conditional stop differ from usual straight-line programs by the following: (i) some first-type operators and, possibly, some inputs of the program are labeled by integers from 0 to m; (ii) an operator labeled by zero terminates the execution of the program if its value is equal to unity; (iii) the ith value of the program after its termination is the value of its last executed operator (or its input if such an operator is absent) labeled by i.

We find upper and lower bounds on the Shannon functions for the average time of computing Boolean operators by first- and second-type programs. Asymptotically exact formulas are derived in the case when the number of components of the operators being computed increases with the number of their arguments.

2. Basic definitions

Below are formal definitions. Let $B \subseteq \{f : \{0,1\}^k \to \{0,1\}\}$ be a basis in P_2, $\pi : \{0,1\}^{m+1} \to \{0,1\}^{m+1}$ be the identity Boolean operator, and $X_n = \{x_1, \ldots, x_n\}$ be a set of independent variables.

The *first-type straight-line program with a conditional stop* is a sequence $P = p_1 \ldots p_i \ldots p_s$ whose elements are the operators $p_i = f_i(p_{i,1}, \ldots, p_{i,l})$, where $p_{i,j} \in \{p_1, \ldots, p_{i-1}\} \cup X_n$, $f_i \in B \cup \{\pi\}$; moreover, if $p_{i,j} = p_t \in \{p_1, \ldots, p_{i-1}\}$, then $f_t \neq \pi$. An operator p_i is called a *first-type* (or *functional*) *operator* if $f_i \neq \pi$. An operator p_i is called a *second-type* (or *stop*) *operator* if $f_i = \pi$. The program P is said to have n inputs and m outputs, and the variable x_i is said to be assigned to its ith input.

Put $n(p_i) = i$, i.e., $n(p)$ is the index of the operator p in the program P. Let $p_{i_1}, \ldots, p_{i_r}$ be all second-type operators in P, $i_1 < \cdots < i_r$. Denote by q_t the tth second-type operator in P and by $q_{t,j}$ the jth argument of this operator. The value of an operator p in P on an arbitrary binary tuple x is defined by induction. We set $p_1(x) = f_1(x)$ for the first operator and $p_i(x) = f_i(p_{i,1}(x), \ldots, p_{i,l}(x))$ for $i > 1$.

The result of P on x is denoted by $P(x) = (P_1(x), \ldots, P_m(x))$, and its jth component is defined as

$$P_j(x) = q_{1,m+1}(x)q_{1,j}(x) \vee \bar{q}_{1,m+1}(x)(q_{2,m+1}(x)q_{2,j}(x) \vee \cdots \vee \bar{q}_{r-2,m+1}(x)(q_{r-1,m+1}(x)q_{r-1,j}(x) \vee \bar{q}_{r-1,m+1}(x)q_{r,m+1}(x)q_{r,j}(x))\ldots).$$

Let $B \subseteq \{f : \{0,1\}^k \to \{0,1\}\}$ be a basis in P_2 and $X_n = \{x_1, \ldots, x_n\}$ be a set of independent variables, each associated with a number $a_i \in \{0, 1, \ldots, m\}$.

The *second-type straight-line program with a conditional stop* is a sequence $P = p_1 \ldots p_i \ldots p_s$ whose elements are the operators $p_i = (f_i(p_{i,1}, \ldots, p_{i,l}), a_i)$, where $a_i \in \{0, 1, \ldots, m\}$, $f_i \in B$, and $p_{i,j} \in \{p_1, \ldots, p_{i-1}\} \cup X_n$. An operator p_i is called a *first-type* (or *functional*) *operator* if $a_i \neq 0$ and a *second-type* (or *stop*) *operator* if $a_i = 0$. The program P is said to have n inputs and m outputs, and the variable x_i is said to be assigned to its ith input.

The value of an operator p of a second-type program P on any binary tuple x is defined in the same manner as for operators of first-type programs. As before, let $n(p)$ be the index of p in P and q_t be the tth second-type operator in P. Let $q_{t,j}$ denote an operator p_i such that $i = \max s$, where the maximum is over all s such that $a_s = j$ and $s < n(q_t)$, unless such an operator is absent, in which case let $q_{t,j}$ denote the variable x_r with a maximum index r to which the number j is assigned. The result of a second-type program P on a tuple x is denoted by $P(x) = (P_1(x), \ldots, P_m(x))$, and the value of its jth component is defined as

$$P_j(x) = q_1(x)q_{1,j}(x) \vee \bar{q}_1(x)(q_2(x)q_{2,j}(x) \vee \cdots$$

$$\vee \bar{q}_{r-2}(x)(q_{r-1}(x)q_{r-1,j}(x) \vee \bar{q}_{r-1}(x)q_r(x)q_{r,j}(x))\ldots).$$

The *execution time* $T_P(x)$ of a first-type program P on a tuple of variables x is the minimum $n(q_j)$ such that $q_{j,m+1}(x) = 1$, i.e., the number of operators executed before the program terminates. The *execution time* $T_P(x)$ of a second-type program P on a tuple of variables x is the minimum $n(q_j)$ such that $q_j(x) = 1$. The quantity

$$T(P) = 2^{-n} \sum T_P(x),$$

where summation is over all binary tuples of length n, is called the *average execution time* of P. If $f(x) = P(x)$ for a Boolean operator f and any binary tuple x, then program P is said to compute the operator f. The quantity

$$T_{B,i}(f) = \min T(P),$$

where the minimum is over all programs of the ith type that compute f over the basis B, is called the *average time of computing* f. A program P computing f in the average time $T(P) = T_{B,i}(f)$ is called *minimal*. The number of elements of a Boolean circuit implementing an operator f over a basis B is called the *complexity* of f and is denoted by $L_B(f)$.

The Shannon functions $T_{B,i}(n,m)$ are defined in the standard way as

$$T_{B,i}(n,m) = \max T_{B,i}(f),$$

where the maximum is over all Boolean (n,m)-operators $f : \{0,1\}^n \to \{0,1\}^m$.

In what follows, we consider a basis consisting of all at most k-place Boolean functions, where k is a constant. The average time of computing f by programs of the ith type over this basis is denoted by $T_{k,i}(f)$, and the corresponding Shannon functions, by $T_{k,i}(n,m)$. Suppose that the number n of arguments of each Boolean operator considered below is sufficiently large and the number m of components of these operators satisfies $m = n^{\mathcal{O}(1)}$. Let c and c_i $(i = 1,2,\ldots)$ denote suitable constants and log denote the logarithm to the base 2. The concepts used without definitions can be found in [3].

3. The main results

The main results of this paper are bounds on Shannon functions. They are given in Theorems 1 and 2. The cases when asymptotically exact formulas can be derived

for Shannon functions are formulated as consequences of these theorems. The upper bounds are proved by conventional methods [3] used in Boolean circuits. The lower bounds are proved by a cardinality method (by comparing the number of programs with the number of Boolean operators).

Set

$$\alpha(m,k)=\begin{cases}\frac{m}{m+k} \text{ if } k<m+2,\\ \frac{m}{2k-2} \text{ if } k\geqslant m+2,\end{cases} \qquad \beta(m,k)=\min\left(1,\frac{m}{2k-2}\right).$$

Theorem 1. *Let k be a constant and $m=n^{\mathcal{O}(1)}$. Then*

$$\alpha(m,k)\frac{2^n}{n}(1+\mathrm{o}(1))\leqslant T_{k,1}(n,m)\leqslant\beta(m,k)\frac{2^n}{n}(1+\mathrm{o}(1)).$$

The lower bound follows from Lemmas 5 and 6 proved below. The upper bound follows from Lemmas 8 and 9.

Theorem 2. *Let k be a constant and $m=n^{\mathcal{O}(1)}$. Then*

$$\frac{2^{n-1}m}{nk}(1+\mathrm{o}(1))\leqslant T_{k,2}(n,m)\leqslant\frac{2^{n-1}(m+1)}{nk}(1+\mathrm{o}(1)).$$

The lower and upper bounds in the theorem follow from Lemmas 7 and 10, respectively.

The next assertions are easily deduced from Theorems 1 and 2.

Corollary 1. *Let k be a constant and $k\geqslant m+2$. Then*

$$T_{k,1}(n,m)=\frac{2^{n-1}m}{n(k-1)}(1+\mathrm{o}(1)).$$

Corollary 2. *Let k be a constant, $m\to\infty$, and $m=n^{\mathcal{O}(1)}$. Then*

$$T_{k,1}(n,m)=\frac{2^n}{n}(1+\mathrm{o}(1)).$$

Corollary 3. *Let k be a constant, $m\to\infty$, and $m=n^{\mathcal{O}(1)}$. Then*

$$T_{k,2}(n,m)=\frac{2^{n-1}m}{nk}(1+\mathrm{o}(1)).$$

4. Bounds for the number of programs

A first-type program $P=p_1\dots p_s$ is said to be *terminal* if for any $i\in\{1,\dots,s\}$ there exists a tuple x such that $P(x)\neq P'(x)$, where $P'=p_1\dots p_{i-1}p_{i+1}\dots p_s$. A second-type

program P is called *reduced* if the values computed by the stop operators are not used as arguments of the other operators.

Obviously, any first-type program can be transformed into a terminal program such that the average execution time of the new program is at most that of the original program. An analogous statement holds for second-type programs: any second-type program can be transformed into a reduced program such that its average execution time is at most the average execution time of the original program.

Indeed, suppose that a second-type program has a stop operator whose value is an argument of another operator. If the value of the stop operator is zero, then it can be replaced by a constant; but if its value equals unity, it can be replaced by zero, because the operator terminates the program and this value is nowhere used.

To each first-type program P with m outputs and with its inputs associated with variables $x_1,\dots,x_n$, we assign a directed graph GP as follows:

(i) a vertex of GP is assigned to each variable x_i and to each operator p_j of P;

(ii) the symbol x_i is assigned to the vertex associated with the variable x_i;

(iii) the symbol f_j is assigned to the vertex associated with the operator p_j;

(iv) vertices u_i and u_j are connected by an arc directed from u_i to u_j if u_j corresponds to an operator p whose sth argument is the operator associated with u_i, $1 \leqslant s \leqslant \max(m+1,k)$; each such arc is assigned the symbol s; and

(v) vertices u_i and u_j are connected by an arc directed from u_i to u_j if they correspond to the lth and $(l+1)$th stop operators of the program P; each such arc is assigned the symbol 0.

To each second-type program P with m outputs and with its inputs associated with variables $x_1,\dots,x_n$, we assign a directed graph GP as follows:

(i) a vertex of GP is assigned to each variable x_i and to each operator p_j of the program P;

(ii) the symbol x_i is assigned to the vertex associated with the variable x_i;

(iii) the symbol f_j is assigned to the vertex associated with the operator p_j;

(iv) vertices u_i and u_j are connected by an arc directed from u_i to u_j if u_j corresponds to the operator p whose sth argument is the operator associated with u_i, $1 \leqslant s \leqslant k$; each such arc is called functional and is assigned the symbol s;

(v) vertices u_i and u_j are connected by an arc directed from u_i to u_j if u_j corresponds to the stop operator q_t, the vertex u_i corresponds to $q_{t,s}$, and $q_{t,s} \neq q_{r,s}$ for all $r < t$ and $1 \leqslant s \leqslant m$; each such arc is called specific and is assigned the symbol s; and

(vi) vertices u_i and u_j are connected by an arc directed from u_i to u_j if they correspond to the lth and $(l+1)$th stop operators of the program P; each such arc is called a stop arc and is assigned the symbol 0.

Programs P_1 and P_2 are said to be *isomorphic* if their corresponding graphs are isomorphic. It is easy to see that any program can be restored, up to an isomorphism, from its corresponding graph.

We introduce the following notation:

$N_1(k,n,m,L_1,L_2)$ is the number of nonisomorphic first-type programs over the basis of all at most k-place Boolean functions containing at most L_1 functional operators and at most L_2 stop operators with n inputs and m outputs;

$N_1(k,n,m,L)$ is the number of nonisomorphic terminal programs of the first type over the basis of all at most k-place Boolean functions containing at most L operators with n inputs and m outputs;

$N_2(k,n,m,L_1,L_2)$ is the number of nonisomorphic second-type programs over the basis of all at most k-place Boolean functions containing at most L_1 functional operators and at most L_2 stop operators with n inputs and m outputs;

$N_2(k,n,m,L)$ is the number of nonisomorphic reduced programs of the second type over the basis of at most k-place Boolean functions containing at most L operators with n inputs and m outputs.

Lemma 1. *If k is a constant, then*

$$N_1(k,n,m,L_1,L_2) \leqslant (c_1 m(L_1+L_2+n))^{(k-1)L_1+(m+1)L_2},$$

$$N_2(k,n,m,L_1,L_2) \leqslant (c_1 m(L_1+L_2+n))^{(k-1)(L_1+L_2)+(m+1)L_2}.$$

Proof. Since both inequalities in the lemma are proved in a similar way, we prove only the first. To this end, it is sufficient to estimate from above the number of graphs corresponding to the programs under consideration. Each graph contains at most L_1+L_2+n vertices and at most $kL_1+(m+2)L_2-1$ arcs. The number of such graphs does not exceed (see, e.g., [1])

$$(c(L_1+L_2+n))^{kL_1+(m+2)L_2-L_1-L_2-n}.$$

The arcs of the graph can be labeled in at most $k^{kL_1}(m+2)^{(m+2)L_2}$ ways, and its vertices, in at most $(L_1+L_2+n)^n(c(k))^{L_1+L_2}$ ways, where $c(k)$ is a constant depending on k. Therefore, the number of distinct labeled graphs does not exceed the product of the three quantities above, i.e.,

$$N_1(k,n,m,L_1,L_2) \leqslant (c_1 m(L_1+L_2+n))^{(k-1)L_1+(m+1)L_2}.$$

Lemma 1 is proved. □

Lemma 2 is a straightforward consequence of Lemma 1 and is presented without proof.

Lemma 2. *If $k \geqslant m+2$, then*

$$N_1(k,n,m,L) \leqslant (c_2 m(L+n))^{(k-1)L}.$$

Lemma 3. *If $k \leqslant m+2$, then*

$$N_1(k,n,m,L) \leqslant (c_3 m(L+n))^{(L+n)(m+k)/2}.$$

Proof. We show that $q_{i,m+1} \neq q_{j,m+1}$ when $i \neq j$ for any terminal program P. Assume that $q_{i,m+1} = q_{j,m+1}$ and $i < j$. If $q_{i,m+1} = 1$, then the operator q_i terminates the program, and the operator q_j is not computed. If $q_{i,m+1} = 0$, then the operator q_j does not terminate

the program P. Consequently, the operator q_j can be removed from the program. Hence, the assumption made contradicts the fact that P is a terminal program. It follows from $q_{i,m+1} \neq q_{j,m+1}$ that $L_2 \leqslant L_1 + n$. Then $L_2 \leqslant (L+n)/2$ and $L_1 = L - L_2$. Therefore,

$$\begin{aligned}
&(k-1)L_1 + (m+1)L_2 \\
&\quad \leqslant (k-1)(L-L_2) + (m+1)L_2 \leqslant (k-1)L + (m+1-k+1)L_2 \\
&\quad \leqslant (2k-2)\frac{L+n}{2} + (m-k+2)\frac{L+n}{2} \\
&\quad \leqslant (m+k)\frac{L+n}{2}.
\end{aligned}$$

Substituting this bound into the first inequality in Lemma 1, we obtain the required result. Lemma 3 is proved. □

Lemma 4. *If k is a constant, then*

$$N_2(k,n,m,L) \leqslant (c_4 m(L+n))^{kL-n}.$$

Proof. We estimate from above the number of graphs corresponding to the programs under consideration. Let L_1 be the number of functional operators and L_2 be the number of stop operators in the program. Obviously, $L = L_1 + L_2$, and each graph contains at most $L+n$ vertices. Let us estimate the number of arcs. Each vertex corresponding to a functional operator emanates at most one specific arc. The number of specific arcs is denoted by N. Obviously, no specific or functional arcs leave the vertices corresponding to the stop operators. Hence, $N \leqslant L_1$. Each graph contains $L_2 - 1$ stop arcs and at most kL functional arcs. Consequently, the total number of arcs is at most $kL + L_2 + N$. It is easy to see that the greater the difference between the numbers of arcs and vertices, the larger the number of graphs is (see [2]). This difference can be transformed to give

$$(k-1)L + L_2 + N - n \leqslant (k-1)L + L_2 + L_1 - n = kL - n.$$

Taking into account the number of ways in which the arcs and vertices of the graph can be labeled, we obtain

$$N_2(k,n,m,L) \leqslant (c_4 m(L+n))^{kL-n}.$$

Lemma 4 is proved. □

5. Lower bounds

Let f be a Boolean operator, and P be a program computing f. To each binary tuple x of length n, which is viewed as the binary representation of a positive integer, we assign its number $N_P(x)$ such that $1 \leqslant N_P(x) \leqslant 2^n$ and $N_P(x) < N_P(y)$ if $T_P(x) < T_P(y)$, and $N_P(x) < N_P(y)$ if $T_P(x) = T_P(y)$ and $x < y$.

Lemma 5. *Let $k < m+2$. Then for almost all Boolean (n,m)-operators f*

$$T_{k,1}(f) \geqslant \frac{2^n m}{n(m+k)}(1+\mathrm{o}(1)).$$

Proof. Let f be a Boolean operator and P be a minimal program computing f. Let x_i be such that $N_P(x_i) = i2^n/q$, where $q = 2^{q_0}$, with q_0 being an integer, $q \asymp n/\log^2 n$, and $i = 2,3,\ldots,q$. We estimate from above the number of Boolean operators that can be implemented by minimal programs containing x_i such that $T_P(x_i) \leqslant ((i-1)2^n/qn)\cdot 2m/(m+k)$. Each such operator is uniquely determined by the first $T_P(x_i)$ operators of its minimal program and by a set of at most $2^n - N_P(x_i)$ binary vectors of length m that are its values on the arguments requiring more execution time than x_i. By Lemma 3, the number N_i of distinct programs whose complexity does not exceed $T_P(x_i)$ satisfies the inequality

$$N_i \leqslant \left(c_3 m\left(\frac{(i-1)2^n}{qn}\,\frac{2m}{m+k}+n\right)\right)^{(((i-1)2^n)/q^n)2m/(m+k)+n)(m+k)/2}.$$

Since $m = n^{\mathcal{O}(1)}$, after some simple algebra, we obtain

$$N_i \leqslant 2^{(i-1)m2^n/q}(1+\mathcal{O}(\log n/n)).$$

Thus, denoting by M the number of operators in question, we have

$$M \leqslant \sum_{i=2}^{q} 2^{(i-1)m2^n/q}(1+\mathcal{O}(\log n/n))2^{m2^n - mi2^n/q}.$$

Taking into account $i \leqslant q \asymp n/\log^2 n$, we transform the exponent of the quantity under the summation sign to obtain

$$\begin{aligned} &m2^n - \frac{mi2^n}{q} + \frac{(i-1)m2^n}{q}\left(1+\mathcal{O}\left(\frac{\log n}{n}\right)\right)\\ &\quad = m2^n - \frac{m2^n}{q}\left(i-(i-1)\left(1+\mathcal{O}\left(\frac{\log n}{n}\right)\right)\right)\\ &\quad = m2^n - \frac{m2^n}{q}\left(1+\mathcal{O}\left(\frac{1}{\log n}\right)\right). \end{aligned}$$

Therefore,

$$M \leqslant q2^{m2^n - m2^n/q(1+\mathcal{O}(1/\log n))} = 2^{m2^n - m2^n/q^{(1+\mathcal{O}(1/\log n))}}.$$

A comparison of this bound for M with the number of all Boolean (n,m)-operators shows that all minimal programs of almost all Boolean operators satisfy the following condition:

$$\begin{aligned} &\text{if } x_i \text{ is such that } N_P(x_i) = \frac{i2^n}{q}, \text{ where } q = 2^{q_0} \ (q_0 \text{ is an integer})\\ &q \asymp \frac{n}{\log^2 n}, \text{ and } i = 2,3,\ldots,q, \text{ then } T_P(x_i) > \frac{(i-1)2^n}{qn}\,\frac{2m}{m+k}. \end{aligned} \tag{1}$$

Set $X_i = \{x \mid N_P(x_i) < N_P(x) \leqslant N_P(x_{i+1})\}$. For the average execution time $T(P)$ of each such program P we then have

$$\begin{aligned} T(P) &= 2^{-n} \sum_x T_P(x) \geqslant 2^{-n} \sum_{i=2}^{q-1} T_P(x_i)|X_i| \\ &= 2^{-n} \sum_{i=2}^{q-1} T_P(x_i) \frac{2^n}{q} > \frac{1}{q} \sum_{i=2}^{q-1} \frac{(i-1)2^n}{qn} \frac{2m}{m+k} \\ &= \frac{(q-1)(q-2)2^n}{q^2 n} \frac{m}{m+k} \\ &= \frac{2^n m}{n(m+k)} \left(1 + \mathcal{O}\left(\frac{\log^2 n}{n}\right)\right). \end{aligned}$$

Lemma 5 is proved. □

Lemma 6. *Let $k \geqslant m+2$. Then for almost all Boolean (n,m)-operators f*

$$T_{k,1}(f) \geqslant \frac{2^{n-1}m}{n(k-1)}(1 + \mathrm{o}(1)).$$

Proof. As in the proof of the preceding lemma, it is easy to show that all minimal programs P that implement almost all Boolean operators satisfy the following condition:

$$\begin{aligned} &\text{if } x_i \text{ is such that } N_P(x_i) = \frac{i2^n}{q}, \text{ where } q = 2^{q_0} \ (q_0 \text{ is an integer}) \\ &q \asymp \frac{n}{\log^2 n}, \text{ and } i = 2,3,\ldots,q, \text{ then } T_P(x_i) > \frac{(i-1)2^n}{qn} \frac{m}{k-1}. \end{aligned} \tag{2}$$

The proof of (2) is analogous to that of (1); the only difference is that Lemma 2 is used instead of Lemma 3. Once again we set $X_i = \{x \mid N_P(x_i) < N_P(x) \leqslant N_P(x_{i+1})\}$. Then, for the average execution time $T(P)$ of the program P satisfying (2), we have

$$\begin{aligned} T(P) &= 2^{-n} \sum_x T_P(x) \geqslant 2^{-n} \sum_{i=2}^{q-1} T_P(x_i)|X_i| \\ &= 2^{-n} \sum_{i=2}^{q-1} T_P(x_i) \frac{2^n}{q} > \frac{1}{q} \sum_{i=2}^{q-1} \frac{(i-1)2^n}{qn} \frac{m}{k-1} \\ &= \frac{(q-1)(q-2)2^n}{2q^2 n} \frac{m}{k-1} \\ &= \frac{2^{n-1}m}{n(k-1)} \left(1 + \mathcal{O}\left(\frac{\log^2 n}{n}\right)\right). \end{aligned}$$

Lemma 6 is proved. □

The next lemma is given without proof, for it almost literally coincides with that of Lemma 5. The only difference is that Lemma 4 is used instead of Lemma 3, and (1) is replaced by the condition

$$\text{if } x_i \text{ is such that } N_P(x_i) = \frac{i2^n}{q}, \text{ where } q = 2^{q_0} \ (q_0 \text{ is an integer})$$

$$q \asymp \frac{n}{\log^2 n}, \text{ and } i = 2, 3, \ldots, q, \text{ then } T_P(x_i) > \frac{(i-1)2^n}{qn}\,\frac{m}{k}.$$

Lemma 7. *For almost all (n,m)-operators f*

$$T_{k,2}(f) \geqslant \frac{2^{n-1}m}{nk}(1 + \mathrm{o}(1)).$$

6. Upper bounds

Let $\sigma = (\sigma_1, \ldots, \sigma_s)$ be a binary tuple. On the set of such tuples, we define the function $N(\sigma) = \sum_{i=1}^{s} \sigma_i 2^{i-1}$.

Lemma 8. *For any Boolean (n,m)-operator f*

$$T_{k,1}(f) \leqslant \frac{2^{n-1}m}{n(k-1)}(1 + \mathrm{o}(1)).$$

Proof. Let $f = (f_1, \ldots, f_m)$, where f_i is the ith component of f. Setting $s = \lfloor \log n \rfloor$ and decomposing f_i in the first s variables, we have

$$f_i(x_1, \ldots, x_n) = \bigvee_{(\sigma_1, \ldots, \sigma_s)} f_i(\sigma_1, \ldots, \sigma_s, x_{s+1}, \ldots, x_n) x_1^{\sigma_1} \ldots x_s^{\sigma_s}$$

$$= \bigvee_{(\sigma_1, \ldots, \sigma_s)} f_{i,j}(x_{s+1}, \ldots, x_n) x_1^{\sigma_1} \ldots x_s^{\sigma_s},$$

where $j = N(\sigma_1, \ldots, \sigma_s)$. The program P computing f is represented as

$$P = P_1 q_1 \ldots P_s q_s.$$

Here, the program P_j computes the functions $f_{1,j}, \ldots, f_{m,j}$, $x_1^{\sigma_1}, \ldots, x_s^{\sigma_s}$ and consists of only functional operators, while the operator q_j terminates the program P if $x_1^{\sigma_1} \ldots x_s^{\sigma_s} = 1$, where $j = N(\sigma_1, \ldots, \sigma_s)$, and declares $f_{i,j}(x_{s+1}, \ldots, x_n)$ to be the ith value of the program. It follows from [3] that

$$L(P_i) \leqslant \frac{2^{n-s}m}{(k-1)(n-s)}(1 + \mathrm{o}(1)).$$

Therefore,

$$T(P) = 2^{-n} \sum_{(\sigma_{s+1},\ldots,\sigma_n)} \sum_{(\sigma_1,\ldots,\sigma_s)} T_P(\sigma_1,\ldots,\sigma_n)$$

$$\leqslant 2^{-n} \sum_{i=1}^{2^s} \frac{m2^{n-s}}{(k-1)(n-s)} 2^{n-s} i(1+\mathrm{o}(1))$$

$$\leqslant 2^{-n} \frac{m2^{2n-2s}}{(k-1)(n-s)} 2^{2s-1}(1+\mathrm{o}(1)) \leqslant \frac{m2^{n-1}}{(k-1)n}(1+\mathrm{o}(1)).$$

Lemma 8 is proved. □

Let s, h be integers. We divide the set of all binary tuples of length s into disjoint sets Y_i such that $\sigma \in Y_i$ if $(i-1)h \leqslant N(\sigma) < ih$. Define the functions

$$g_l(x_1,\ldots,x_s) = \bigvee_{\sigma \in Y_l} x_1^{\sigma_1} \ldots x_s^{\sigma_s},$$

$$g_{j,l}(x_1,\ldots,x_n) = g_l(x_1,\ldots,x_s) x_{s+1}^{\sigma_{s+1}} \ldots x_n^{\sigma_n},$$

where $N(\sigma_{s+1},\ldots,\sigma_n) = \sum_{i=s+1}^{n} \sigma_i 2^{i-s-1} = j$. Let $z(x_1,\ldots,x_n)$ be an arbitrary Boolean function. Set $z_j(x_1,\ldots,x_s) = z(x_1,\ldots,x_s,\sigma_{s+1},\ldots,\sigma_n)$, where $N(\sigma_{s+1},\ldots,\sigma_n) = j$, and $z_{j,l}(x_1,\ldots,x_s) = z_j(x_1,\ldots,x_s) g_l\ (x_1,\ldots,x_s)$. Let $t = \lceil 2^s/h \rceil$. Then [3] yields

$$z(x_1,\ldots,x_n) = \bigvee_{(\sigma_{s+1},\ldots,\sigma_n)} \left(\bigvee_{l=1}^{t} z_{j,l}(x_1,\ldots,x_s) \right) x_{s+1}^{\sigma_{s+1}} \ldots x_n^{\sigma_n},$$

where $N(\sigma_{s+1},\ldots,\sigma_n) = j$. It is easy to see that

$$z_{j,l}(x_1,\ldots,x_s) x_{s+1}^{\sigma_{s+1}} \ldots x_n^{\sigma_n} = z_{j,l}(x_1,\ldots,x_s) g_{j,l}(x_1,\ldots,x_n).$$

Consequently,

$$z(x_1,\ldots,x_n) = \bigvee_{(\sigma_{s+1},\ldots,\sigma_n)} \left(\bigvee_{l=1}^{t} z_{j,l}(x_1,\ldots,x_s) g_{j,l}(x_1,\ldots,x_n) \right). \tag{3}$$

Let R be the number of distinct functions $z_{j,l}(x_1,\ldots,x_s)$ and L_z be the circuit complexity of these functions. It is easy to see that $R \leqslant 2^{h+s+1}/h$ and

$$L_z \leqslant 2^{h+s+1}. \tag{4}$$

Lemma 9. *For any Boolean (n,m)-operator f*

$$T_{k,1}(f) \leqslant \frac{2^n}{n}(1+\mathrm{o}(1)).$$

Proof. We prove the lemma for $m = 1$. In the general case, the proof differs from that presented below only by additional indices in formulas. Let $z(x_1,\ldots,x_n)$ be an arbitrary

Boolean function. Setting $s = \lfloor 2\log n \rfloor$ and $h = \lfloor n - 4\log n \rfloor$, we use (3). A program computing z can be represented as

$$P = P_1 p_{1,1} q_{1,1} \dots p_{j,l} q_{j,l} \dots p_{t,d} q_{t,d},$$

where $t = \lceil 2^s/h \rceil$, $d = 2^{n-s}$, and P_1 is a program computing all possible functions $z_{j,l}(x_1,\dots,x_s)$, all functions $g_l(x_1,\dots,x_s)$, and all products $x_{s+1}^{\sigma_{s+1}} \dots x_n^{\sigma_n}$. Furthermore, the operator $p_{j,l}$ computes the function $g_{j,l}(x_1,\dots,x_n)$, while the operator $q_{j,l}$ terminates the program if $p_{j,l}=1$ and declares that its result is the value of $z_{j,l}(x_1,\dots,x_s)$ computed by P_1. It is easy to see that

$$T_P(\sigma_1,\dots,\sigma_s,x_{s+1},\dots,x_n) = T_P(\sigma'_1,\dots,\sigma'_s,x_{s+1},\dots,x_n)$$

for all $x_{s+1},\dots,x_n$ whenever $(\sigma_1,\dots,\sigma_s)$ and $(\sigma'_1,\dots,\sigma'_s)$ belong to the same set Y_l. Since $2^{h+s}h^{-1} \leqslant 2^n n^{-2}$, it follows from (3) and (4) that

$$\begin{aligned}
T(P) &= 2^{-n} \sum_{(\sigma_{s+1},\dots,\sigma_n)} \sum_{(\sigma_1,\dots,\sigma_s)} T_P(\sigma_1,\dots,\sigma_n) \\
&= 2^{-n} \sum_{(\sigma_{s+1},\dots,\sigma_n)} \sum_{l=1}^{t} \sum_{(\sigma_1,\dots,\sigma_s)\in Y_l} T_P(\sigma_1,\dots,\sigma_n) \\
&\leqslant 2^{-n} \sum_{(\sigma_{s+1},\dots,\sigma_n)} \sum_{l=1}^{t} h(L_z + 2(N(\sigma_{s+1},\dots,\sigma_n)t + l)) \\
&\leqslant 2^{-n} \left(L_z htd + \sum_{j=1}^{d} \sum_{l=1}^{t} (2h((j-1)t + l) \right) \\
&\leqslant 2^{-n} \left(2^{n+1} 2^n n^{-2} + 2h \sum_{i=1}^{dt} i \right) \leqslant 2^{-n}(2^{2n+1} n^{-2} + h(td)^2) \\
&\leqslant 2^{-n}(2^{2n+1} n^{-2} + h 2^{2n} h^{-2}(1 + \mathrm{o}(1))) \leqslant 2^n n^{-1}(1 + \mathrm{o}(1)).
\end{aligned}$$

Finally, we note that any Boolean operator can be computed by a program that differs from that described above only by the number of arguments in the stop operator. Therefore, the average execution time of the first-type program thus constructed is independent of the number of its arguments. Lemma 9 is proved. □

Lemma 10. *For any Boolean* (n,m)*-operator* f

$$T_{k,2}(f) \leqslant \frac{2^{n-1}(m+1)}{nk}(1 + \mathrm{o}(1)).$$

Proof. Setting $s=\lfloor 2\log n \rfloor$, $h=\lfloor n-4\log n \rfloor$, and $t=\lceil 2^s/h \rceil$, we use (3). Each component f_i of f can be represented as

$$f_i(x_1,\dots,x_n) = \bigvee_{(\sigma_{s+1},\dots,\sigma_n)} \left(\bigvee_{l=1}^{t} f_{i,j,l}(x_1,\dots,x_s) g_{j,l}\,(x_1,\dots,x_n) \right),$$

where $N(\sigma_{s+1},\dots,\sigma_n)=j$. We divide the set of all binary tuples of length s into disjoint subsets $Y_{l'}$ such that $\sigma\in Y_{l'}$ if $(l'-1)kh\leqslant N(\sigma)<l'hk$. The functions $f_{i,j,l}$ are collected in groups $\hat{Y}_{j,l'}$ such that each of them, except possibly the last, consists of k functions, and $f_{i,j,l}\in\hat{Y}_{j,l'}$ if $(l'-1)k\leqslant l<l'k$. Let $w=\lceil t/k\rceil$. Put

$$\hat{g}_{l'}(x_1,\dots,x_s)=\bigvee_{r=1}^{k} g_{(l'-1)k+r}(x_1,\dots,x_s),$$
$$\hat{g}_{j,l'}(x_1,\dots,x_n)=\hat{g}_{l'}(x_1,\dots,x_s)x_{s+1}^{\sigma_{s+1}}\dots x_n^{\sigma_n},$$

where $N(\sigma_{s+1},\dots,\sigma_n)=j$. Since $f_{i,j,l}g_{j,q}=0$ for $l\neq q$, we have

$$f_i(x_1,\dots,x_n)=\bigvee_{(\sigma_{s+1},\dots,\sigma_n)}\left(\bigvee_{l'=1}^{w}\left(\bigvee_{r=1}^{k} f_{i,j,(l'-1)k+r}(x_1,\dots,x_s)\right)\hat{g}_{j,l'}(x_1,\dots,x_n)\right).$$

A program computing the operator f can be represented as

$$P=P_0P_1\dots P_j\dots P_{2^{n-s}},$$

where P_0 is a program computing all the functions $f_{i,j,l}(x_1,\dots,x_s)$ and all the functions $\hat{g}_{l'}(x_1,\dots,x_s)$, while P_j is a program having the form

$$P_j=P_{j,1}\dots P_{j,l'}\dots P_{j,w}$$

where $P_{j,l'}$ is a program that computes $\hat{g}_{j,l'}(x_1,\dots,x_n)$ and all functions $\hat{f}_{i,j,l'}=\bigvee_{r=1}^{k} f_{i,j,(l'-1)k+r}(x_1,\dots,x_s)$, terminates the program if $\hat{g}_{j,l'}(x_1,\dots,x_n)=1$, and declares that the value of $\hat{f}_{i,j,l'}$ is the value of the ith component of f. It is easy to see that each program $P_{j,l'}$ consists of at most $m+1$ operators: a 2-input stop operator computes the function $\hat{g}_{j,l'}(x_1,\dots,x_n)$ and m k-input operators compute the functions $\hat{f}_{i,j,l'}$. Obviously, the circuit complexity L_z of all possible functions $f_{i,j,l}$ and of the functions $\hat{g}_{l'}(x_1,\dots,x_s)$ satisfies (4). It is also easy to see that $T_P(\sigma_1,\dots,\sigma_s,x_{s+1},\dots,x_n)=T_P(\sigma'_1,\dots,\sigma'_s,x_{s+1},\dots,x_n)$ for all $(x_{s+1},\dots,x_n)$ whenever the tuples $(\sigma_1,\dots,\sigma_s)$ and $(\sigma'_1,\dots,\sigma'_s)$ belong to the same set $Y_{l'}$. Setting $d=2^{n-s}$, we have

$$\begin{aligned}T(P)&=2^{-n}\sum_{(\sigma_{s+1},\dots,\sigma_n)}\sum_{(\sigma_1,\dots,\sigma_s)}T_P(\sigma_1,\dots,\sigma_n)\\&=2^{-n}\sum_{(\sigma_{s+1},\dots,\sigma_n)}\sum_{l'=1}^{w}\sum_{(\sigma_1,\dots,\sigma_s)\in Y_{l'}}T_P(\sigma_1,\dots,\sigma_n)\\&\leqslant 2^{-n}\sum_{j=1}^{d}\sum_{l'=1}^{w}(L_z+((j-1)w+l')(m+1))hk\\&\leqslant 2^{-n}\left(L_z dwhk+\sum_{j=1}^{d}\sum_{l'=1}^{w}((j-1)w+l')(m+1)hk\right)\end{aligned}$$

$$\leqslant 2^{-n}\left(L_z dwhk + \frac{(m+1)hk(dw)^2}{2}\right)$$

$$\leqslant 2^{-n}\left(\frac{2^{2n+1}}{n^2} + \frac{(m+1)hk2^{2n}(hk)^{-2}}{2}\right)(1+\mathrm{o}(1))$$

$$\leqslant \frac{2^{n-1}(m+1)}{nk}(1+\mathrm{o}(1)).$$

Lemma 10 is proved. □

7. Programs with a limited number of stop operators

We present without proofs some results concerning the average execution time of programs with a limited number of conditional stops. Consider the first-type programs in which the number of stop operators is $\mathrm{o}(2^n/mn)$ and the second-type programs in which the number of stop operators is $\mathrm{o}(2^n/n)$. Their Shannon functions are denoted by $T'_{k,1}$ and $T'_{k,2}$, respectively.

Theorem 3. *Let k be a constant. Then*

$$T'_{k,1}(n,m) = \frac{m2^{n-1}}{(k-1)n}(1+\mathrm{o}(1)),$$

$$T'_{k,2}(n,m) = \frac{m2^{n-1}}{(k-1)n}(1+\mathrm{o}(1)).$$

The upper bounds are easily proved using Lemma 8, and the lower bounds are proved in the same fashion as Lemma 5, with the number of programs estimated using Lemma 1.

Acknowledgements

The author is grateful to Professor O.B. Lupanov for his attention to this study and to R.M. Kolpakov, who indicated some inaccuracy in the first version of this paper.

References

[1] A.V. Chashkin, Average case complexity for finite boolean functions, Discrete Appl. Math. 114 (2001) 43–59.

[2] O.B. Lupanov, Asymptotic bounds for the number of graphs and networks with n edges, Problemy kibernetiki 4 (1960) 61–80 (in Russian).

[3] O.B. Lupanov, Synthesis of some classes of control systems, Problemy kibernetiki 9 (1963) 63–97 (in Russian).